Part IB — Optimisation

Based on lectures by F. A. Fischer

Notes taken by Dexter Chua

Easter 2015

These notes are not endorsed by the lecturers, and I have modified them (often significantly) after lectures. They are nowhere near accurate representations of what was actually lectured, and in particular, all errors are almost surely mine.

Lagrangian methods
General formulation of constrained problems; the Lagrangian sufficiency theorem. Interpretation of Lagrange multipliers as shadow prices. Examples. [2]

Linear programming in the nondegenerate case
Convexity of feasible region; sufficiency of extreme points. Standardization of problems, slack variables, equivalence of extreme points and basic solutions. The primal simplex algorithm, artificial variables, the two-phase method. Practical use of the algorithm; the tableau. Examples. The dual linear problem, duality theorem in a standardized case, complementary slackness, dual variables and their interpretation as shadow prices. Relationship of the primal simplex algorithm to dual problem. Two person zero-sum games. [6]

Network problems
The Ford-Fulkerson algorithm and the max-flow min-cut theorems in the rational case. Network flows with costs, the transportation algorithm, relationship of dual variables with nodes. Examples. Conditions for optimality in more general networks; *the simplex-on-a-graph algorithm*. [3]

Practice and applications
Efficiency of algorithms. The formulation of simple practical and combinatorial problems as linear programming or network problems. [1]

Contents

1 Introduction and preliminaries **3**
1.1 Constrained optimization 3
1.2 Review of unconstrained optimization 4

2 The method of Lagrange multipliers **6**
2.1 Complementary Slackness 9
2.2 Shadow prices 9
2.3 Lagrange duality 10
2.4 Supporting hyperplanes and convexity 11

3 Solutions of linear programs **14**
3.1 Linear programs 14
3.2 Basic solutions 15
3.3 Extreme points and optimal solutions 16
3.4 Linear programming duality 17
3.5 Simplex method 20
3.5.1 The simplex tableau 21
3.5.2 Using the Tableau 22
3.6 The two-phase simplex method 23

4 Non-cooperative games **26**
4.1 Games and Solutions 26
4.2 The minimax theorem 28

5 Network problems **30**
5.1 Definitions 30
5.2 Minimum-cost flow problem 30
5.3 The transportation problem 31
5.4 The maximum flow problem 36

1 Introduction and preliminaries

1.1 Constrained optimization

In optimization, the objective is to maximize or minimize some function. For example, if we are a factory, we want to minimize our cost of production. Often, our optimization is not unconstrained. Otherwise, the way to minimize costs is to produce nothing at all. Instead, there are some constraints we have to obey. The is known as *constrained optimization*.

Definition (Constrained optimization)**.** The general problem is of *constrained optimization* is

$$\text{minimize } f(x) \text{ subject to } h(x) = b,\ x \in X$$

where $x \in \mathbb{R}^n$ is the *vector of decision variables*, $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, $h : \mathbb{R}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^m$ are the *functional constraints*, and $X \subseteq \mathbb{R}^n$ is the *regional constraint*.

Note that everything above is a vector, but we do not bold our vectors. This is since almost everything we work with is going to be a vector, and there isn't much point in bolding them.

This is indeed the most general form of the problem. If we want to maximize f instead of minimize, we can minimize $-f$. If we want our constraints to be an inequality in the form $h(x) \geq b$, we can introduce a *slack variable* z, make the functional constraint as $h(x) - z = b$, and add the regional constraint $z \geq 0$. So all is good, and this is in fact the most general form.

Linear programming is, surprisingly, the case where everything is linear. We can write our problem as:

$$\begin{gathered}\text{minimize } c^T x \text{ subject to} \\ a_i^T x \geq b_i \text{ for all } i \in M_1 \\ a_i^T x \leq b_i \text{ for all } i \in M_2 \\ a_i^T x = b_i \text{ for all } i \in M_3 \\ x_i \geq 0 \text{ for all } i \in N_1 \\ x_j \leq 0 \text{ for all } i \in N_2\end{gathered}$$

where we've explicitly written out the different forms the constraints can take.

This is too clumsy. Instead, we can perform some tricks and turn them into a nicer form:

Definition (General and standard form)**.** The *general form* of a linear program is

$$\text{minimize } c^T x \text{ subject to } Ax \geq b,\ x \geq 0$$

The *standard form* is

$$\text{minimize } c^T x \text{ subject to } Ax = b,\ x \geq 0.$$

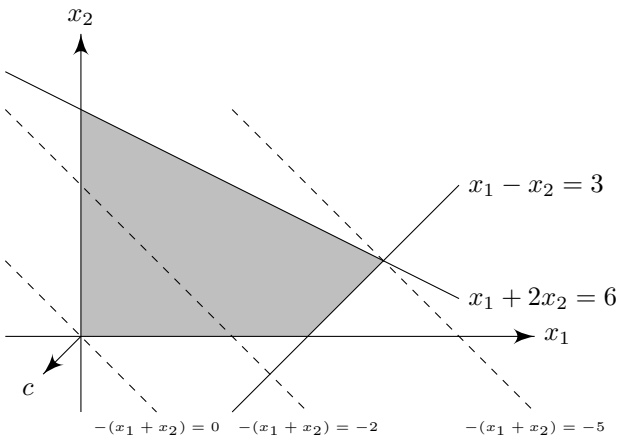
It takes some work to show that these are indeed the most general forms. The equivalence between the two forms can be done via slack variables, as described above. We still have to check some more cases. For example, this form says that $x \geq 0$, i.e. all decision variables have to be positive. What if we want x to be unconstrained, ie can take any value we like? We can split x into to parts, $x = x^+ - x^-$, where each part has to be positive. Then x can take any positive or negative value.

Note that when I said "nicer", I don't mean that turning a problem into this form necessarily makes it easier to solve *in practice*. However, it will be much easier to work with when developing general theory about linear programs.

Example. We want to minimize $-(x_1 + x_2)$ subject to

$$\begin{aligned} x_1 + 2x_2 &\leq 6 \\ x_1 - x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Since we are lucky to have a 2D problem, we can draw this out.



The shaded region is the feasible region, and c is our *cost vector*. The dotted lines, which are orthogonal to c are lines in which the objective function is constant. To minimize our objective function, we want the line to be as right as possible, which is clearly achieved at the intersection of the two boundary lines.

Now we have a problem. In the general case, we have absolutely *no idea* how to solve it. What we *do* know, is how to do *un*constrained optimization.

1.2 Review of unconstrained optimization

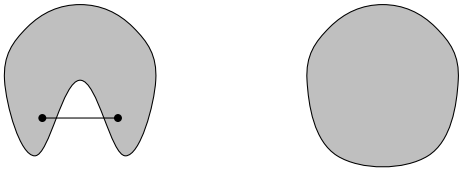
Let $f : \mathbb{R}^n \to \mathbb{R}$, $x^* \in \mathbb{R}^n$. A necessary condition for x^* to minimize f over $\mathbb{R}^n$ is $\nabla f(x^*) = 0$, where

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n} \right)^T$$

is the gradient of f.

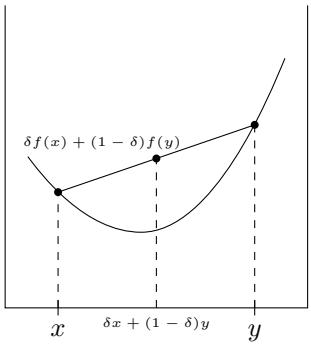
However, this is obviously not a sufficient condition. Any such point can be a maximum, minimum or a saddle. Here we need a notion of convexity:

Definition (Convex region)**.** A region $S \subseteq \mathbb{R}^n$ is *convex* iff for all $\delta \in [0, 1]$, $x, y \in S$, we have $\delta x + (1 - \delta)y \in S$. Alternatively, If you take two points, the line joining them lies completely within the region.



non-convex convex

Definition (Convex function)**.** A function $f : S \to \mathbb{R}$ is *convex* if S is convex, and for all $x, y \in S$, $\delta \in [0, 1]$, we have $\delta f(x) + (1 - \delta)f(y) \geq f(\delta x + (1 - \delta)y)$.



A function is *concave* if $-f$ is convex. Note that a function can be neither concave nor convex.

We have the following lemma:

Lemma. Let f be twice differentiable. Then f is convex on a convex set S if the Hessian matrix

$$Hf_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

is positive semidefinite for all $x \in S$, where this fancy term means:

Definition (Positive-semidefinite)**.** A matrix H is *positive semi-definite* if $v^T H v \geq 0$ for all $v \in \mathbb{R}^n$.

Which leads to the following theorem:

Theorem. Let $X \subseteq \mathbb{R}^n$ be convex, $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable on X. If $x^* \in X$ satisfy $\nabla f(x^*) = 0$ and $Hf(x)$ is positive semidefinite for all $x \in X$, then x^* minimizes f on X.

We will not prove these.

Note that this is helpful, since linear functions are convex (and concave). The problem is that our problems are constrained, not unconstrained. So we will have to convert constrained problems to unconstrained problems.

2 The method of Lagrange multipliers

So how do we solve the problem of constrained maximization? The trick here is to include the constraints into the constraints into the objective function, so that things outside the constraint will not be thought to be minima.

Suppose the original problem is

$$\text{minimize } f(x) \text{ subject to } h(x) = b,\ x \in X.$$

Call the constraint (P).

Definition (Lagrangian)**.** The *Lagrangian* of a constraint (P) is defined as

$$L(x, \lambda) = f(x) - \lambda^T (h(x) - b).$$

for $\lambda \in \mathbb{R}^m$. λ is known as the *Lagrange multiplier*.

Note that when the constraint is satisfied, $h(x) - b = 0$, and $L(x, \lambda) = f(x)$.

We could as well have used

$$L(x, \lambda) = f(x) + \lambda^T (h(x) - b).$$

since we just have to switch the sign of λ. So we don't have to worry about getting the sign of λ wrong when defining the Lagrangian.

If we minimize L over both x and λ, then we will magically find the minimal solution subject to the constrains. Sometimes.

Theorem (Lagrangian sufficiency)**.** Let $x^* \in X$ and $\lambda^* \in \mathbb{R}^m$ be such that

$$L(x^*, \lambda^*) = \inf_{x \in X} L(x, \lambda^*) \quad \text{and} \quad h(x^*) = b.$$

Then x^* is optimal for (P).

In words, if x^* minimizes L for a fixed λ^*, and x^* satisfies the constraints, then x^* minimizes f.

This looks like a pretty powerful result, but it turns out that it is quite easy to prove.

Proof. We first define the "feasible set": let $X(b) = \{x \in X : h(x) = b\}$, i.e. the set of all x that satisfies the constraints. Then

$$\begin{aligned}
\min_{x \in X(b)} f(x) &= \min_{x \in X(b)} (f(x) - \lambda^{*T}(h(x) - b)) \quad \text{since } h(x) - b = 0 \\
&\geq \min_{x \in X} (f(x) - \lambda^{*T}(h(x) - b)) \\
&= f(x^*) - \lambda^{*T}(h(x^*) - b). \\
&= f(x^*).
\end{aligned}$$

□

How can we interpret this result? To find these values of λ^* and x^*, we have to solve

$$\begin{aligned}
\nabla L &= 0 \\
h(x) &= b.
\end{aligned}$$

Alternatively, we can write this as

$$\nabla f = \lambda \nabla h$$
$$h(x) = b.$$

What does this mean? For better visualization, we take the special case where f and h are a functions $\mathbb{R}^2 \to \mathbb{R}$. Usually, if we want to minimize f without restriction, then for small changes in x, there should be no (first-order) change in f, i.e. $\mathrm{d}f = \nabla f \cdot \mathrm{d}x = 0$. This has to be true for all possible directions of x.

However, if we are constrained by $h(x) = b$, this corresponds to forcing x to lie along this particular path. Hence the restriction $\mathrm{d}f = 0$ only has to hold when x lies along the path. Since we need $\nabla f \cdot \mathrm{d}x = 0$, this means that ∇f has to be perpendicular to $\mathrm{d}x$. Alternatively, ∇f has to be parallel to the normal to the path. Since the normal to the path is given by ∇h, we obtain the requirement $\nabla f = \lambda \nabla h$.

This is how we should interpret the condition $\nabla f = \lambda \nabla h$. Instead of requiring that $\nabla f = 0$ as in usual minimization problems, we only require ∇f to point at directions perpendicular to the allowed space.

Example. Minimize $x_1 + x_2 - 2x_3$ subject to

$$x_1 + x_2 + x_3 = 5$$
$$x_1^2 + x_2^2 = 4$$

The Lagrangian is

$$\begin{aligned} L(x, \lambda) &= x_1 - x_2 - 2x_3 - \lambda_1(x_1 + x_2 + x_3 - 5) - \lambda_2(x_1^2 + x_2^2 - 4) \\ &= ((1 - \lambda_1)x_1 - 2\lambda_2 x_1^2) + ((-1 - \lambda_1)x_2 - \lambda_2 x_2^2) \\ &\quad + (-2 - \lambda_1)x_3 + 5\lambda_1 + 4\lambda_2 \end{aligned}$$

We want to pick a λ^* and x^* such that $L(x^*, \lambda^*)$ is minimal. Then in particular, for our λ^*, $L(x, \lambda^*)$ must have a finite minimum.

We note that $(-2 - \lambda_1)x_3$ does not have a finite minimum unless $\lambda_1 = -2$, since x_3 can take any value. Also, the terms in x_1 and x_2 do not have a finite minimum unless $\lambda_2 < 0$.

With these in mind, we find a minimum by setting all first derivatives to be 0:

$$\frac{\partial L}{\partial x_1} = 1 - \lambda_1 - 2\lambda_2 x_1 = 3 - 2\lambda_2 x_1$$
$$\frac{\partial L}{\partial x_2} = -1 - \lambda_1 - 2\lambda_2 x_2 = 1 - 2\lambda_2 x_2$$

Since these must be both 0, we must have

$$x_1 = \frac{3}{2\lambda_2}, \quad x_2 = \frac{1}{2\lambda_2}.$$

To show that this is indeed a minimum, we look at the Hessian matrix:

$$HL = \begin{pmatrix} -2\lambda_2 & 0 \\ 0 & -2\lambda_2 \end{pmatrix}$$

which is positive semidefinite when $\lambda_2 < 0$, which is the condition we came up with at the beginning.

Let $Y = \{\lambda : \mathbb{R}^2 : \lambda_1 = -2, \lambda_2 < 0\}$ be our helpful values of λ.

So we have shown above that for every $\lambda \in Y$, $L(x, \lambda)$ has a unique minimum at $x(\lambda) = (\frac{3}{2\lambda_2}, \frac{1}{2\lambda 2}, x_3)^T$.

Now all we have to do is find λ and x such that $x(\lambda)$ satisfies the functional constraints. The second constraint gives

$$x_1^2 + x_2^2 = \frac{9}{4\lambda^2} + \frac{1}{4\lambda_2^2} = 4 \Leftrightarrow \lambda_2 = -\sqrt{\frac{5}{8}}.$$

The first constraint gives

$$x_3 = 5 - x_1 - x_2.$$

So the theorem implies that

$$x_1 = -3\sqrt{\frac{2}{5}}, \quad x_2 = -\sqrt{\frac{2}{5}}, \quad x_3 = 5 + 4\sqrt{\frac{2}{5}}.$$

So far so good. But what if our functional constraint is an inequality? We will need slack variables.

To minimize $f(x)$ subject to $h(x) \leq b$, $x \in X$, we proceed as follows:

(i) Introduce slack variables to obtain the equivalent problem, to minimize $f(x)$ subject to $h(x) + z = b$, $x \in X$, $z \geq 0$.

(ii) Compute the Lagrangian

$$L(x, z, \lambda) = f(x) - \lambda^T(f(x) + z - b).$$

(iii) Find

$$Y = \left\{\lambda : \inf_{x \in X, z \geq 0} L(x, z, \lambda) > -\infty\right\}.$$

(iv) For each $\lambda \in Y$, minimize $L(x, z, \lambda)$, i.e. find

$$x^*(\lambda) \in X, \quad z^*(\lambda) \geq 0$$

such that

$$L(x^*(\lambda), z^*(\lambda), \lambda) = \inf_{x \in X, z \geq 0} L(x, z, \lambda)$$

(v) Find $\lambda^* \in Y$ such that

$$h(x^*(\lambda^*)) + z^*(\lambda^*) = b.$$

Then by the Lagrangian sufficiency condition, $x^*(\lambda^*)$ is optimal for the constrained problem.

2.1 Complementary Slackness

If we introduce a slack variable z, we note that changing the value of z_j does not affect our objective function, and we are allowed to pick any positive z. Hence if the corresponding Lagrange multiplier is λ_j, then we must have $(z^*(\lambda))_j\lambda_j = 0$. This is since by definition $z^*(\lambda)_j$ minimizes $z_j\lambda_j$. Hence if $z_j\lambda_j \neq 0$, we can tweak the values of z_j to make a smaller $z_j\lambda_j$.

This makes our life easier since our search space is smaller.

Example. Consider the following problem:

$$\text{maximize } x_1 - 3x_2 \text{ subject to}$$
$$\begin{aligned} x_1^2 + x_2^2 + z_1 &= 4 \\ x_1 + x_2 + z_2 + z_2 &= 2 \\ z_1, z_2 &\geq 0. \end{aligned}$$

where z_1, z_2 are slack variables.

The Lagrangian is

$$L(x, z, \lambda) = ((1-\lambda_2)x_1 - \lambda_1 x_1^2) + ((-3-\lambda_2)x_2 - \lambda_1 x_2^2) - \lambda_1 z_1 - \lambda_2 z_2 + 4\lambda_1 + 2\lambda_2.$$

To ensure finite minimum, we need $\lambda_1, \lambda_2 \leq 0$.

By complementary slackness, $\lambda_1 z_1 = \lambda_2 z_2 = 0$. We can then consider the cases $\lambda_1 = 0$ and $z_1 = 0$ separately, and save a lot of algebra.

2.2 Shadow prices

We have previously described how we can understand the requirement $\nabla f = \lambda \nabla h$. But what does the multiplier λ represent?

Theorem. Consider the problem

$$\text{minimize } f(x) \text{ subject to } h(x) = b.$$

Here we assume all functions are continuously differentiable. Suppose that for each $b \in \mathbb{R}^n$, $\phi(b)$ is the optimal value of f and λ^* is the corresponding Lagrange multiplier. Then

$$\frac{\partial \phi}{\partial b_i} = \lambda_i^*.$$

Proof is omitted, as it is just a tedious application of chain rule etc.

This can be interpreted as follows: suppose we are a factory which is capable of producing m different kinds of goods. Since we have finitely many resources, and producing stuff requires resources, $h(x) = b$ limits the amount of goods we can produce. Now of course, if we have more resources, i.e. we change the value of b, we will be able to produce more/less stuff, and thus generate more profit. The change in profit per change in b is given by $\frac{\partial \phi}{\partial b_i}$, which is the value of λ.

The result also holds when the functional constraints are inequality constraints. If the ith constraint holds with equality at the optimal solution, then the above reasoning holds. Otherwise, if it is not held with equality, then the Lagrange multiplier is 0 by complementary slackness. Also, the partial derivative of ϕ with respect to b_i also has to be 0, since changing the upper bound doesn't affect us if we are not at the limit. So they are equal.

2.3 Lagrange duality

Consider the problem

$$\text{minimize } f(x) \text{ subject to } h(x) = b,\ x \in X.$$

Denote this as P.

The Lagrangian is

$$L(x, \lambda) = f(x) - \lambda^T (h(x) - b).$$

Define the dual function $g : \mathbb{R}^m \to \mathbb{R}$ as

$$g(\lambda) = \inf_{x \in X} L(x, \lambda).$$

ie, we fix λ, and see how small we can get L to be. As before, let

$$Y = \{\lambda \in \mathbb{R}^n : g(\lambda) > -\infty\}.$$

Then we have

Theorem (Weak duality)**.** If $x \in X(b)$ (i.e. x satisfies both the functional and regional constraints) and $\lambda \in Y$, then

$$g(\lambda) \leq f(x).$$

In particular,

$$\sup_{\lambda \in Y} g(\lambda) \leq \inf_{x \in X(b)} f(x).$$

Proof.

$$\begin{aligned} g(\lambda) &= \inf_{x' \in X} L(x', \lambda) \\ &\leq L(x, \lambda) \\ &= f(x) - \lambda^T (h(x) - b) \\ &= f(x). \end{aligned}$$

□

This suggests that we can solve a dual problem - instead of minimizing f, we can maximize g subject to $\lambda \in Y$. Denote this problem as (D). The original problem (P) is called *primal*.

Definition (Strong duality)**.** (P) and (D) are said to satisfy *strong duality* if

$$\sup_{\lambda \in Y} g(\lambda) = \inf_{x \in X(b)} f(x).$$

It turns out that problems satisfying strong duality are exactly those for which the method of Lagrange multipliers work.

Example. Again consider the problem to minimize $x_1 - x_2 - 2x_3$ subject to

$$\begin{aligned} x_1 + x_2 + x_3 &= 5 \\ x_1^2 + x_2^2 &= 4 \end{aligned}$$

We saw that

$$Y = \{\lambda \in \mathbb{R}^2 : \lambda_1 = -2, \lambda_2 < 0\}$$

and

$$x^*(\lambda) = \left(\frac{3}{2\lambda_2}, \frac{1}{2\lambda_2}, 5 - \frac{4}{2\lambda_2}\right).$$

The dual function is

$$g(\lambda) = \inf_{x \in X} L(x, \lambda) = L(x^*(\lambda), \lambda) = \frac{10}{4\lambda_2} + 4\lambda_2 - 10.$$

The dual is the problem to

$$\text{maximize } \tfrac{10}{4\lambda_2} + 4\lambda_2 - 10 \text{ subject to } \lambda_2 < 0.$$

The maximum is attained for

$$\lambda_2 = -\sqrt{\frac{5}{8}}$$

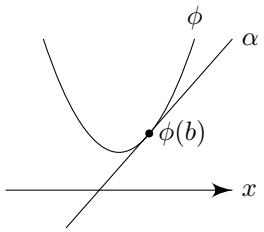
After calculating the values of g and f, we can see that the primal and dual do have the same optimal value.

Right now, what we've got isn't helpful, because we won't know if our problem satisfies strong duality!

2.4 Supporting hyperplanes and convexity

We use the fancy term "hyperplane" to denote planes in higher dimensions (in an n-dimensional space, a hyperplane has $n - 1$ dimensions).

Definition (Supporting hyperplane)**.** A hyperplane $\alpha : \mathbb{R}^m \to \mathbb{R}$ is *supporting* to ϕ at b if α intersects ϕ at b and $\phi(c) \geq \alpha(c)$ for all c.



Theorem. (P) satisfies strong duality iff $\phi(c) = \inf_{x \in X(c)} f(x)$ has a supporting hyperplane at b.

Note that here we fix a b, and let ϕ be a function of c.

Proof. ($\Leftarrow$) Suppose there is a supporting hyperplane. Then since the plane passes through $\phi(b)$, it must be of the form

$$\alpha(c) = \phi(b) + \lambda^T(c - b).$$

Since this is supporting, for all $c \in \mathbb{R}^m$,

$$\phi(b) + \lambda^T(c - b) \leq \phi(c),$$

or

$$\phi(b) \leq \phi(c) - \lambda^T(c - b),$$

This implies that

$$\begin{aligned}\phi(b) &\leq \inf_{c\in\mathbb{R}^m}(\phi(c) - \lambda^T(c-b)) \\ &= \inf_{c\in\mathbb{R}^m}\inf_{x\in X(c)}(f(x) - \lambda^T(h(x)-b))\end{aligned}$$

(since $\phi(c) = \inf_{x\in X(c)} f(x)$ and $h(x) = c$ for $x \in X(c)$)

$$= \inf_{x\in X} L(x,\lambda).$$

(since $\bigcup_{c\in\mathbb{R}^m} X(c) = X$, which is true since for any $x \in X$, we have $x \in X(h(x))$)

$$= g(\lambda)$$

By weak duality, $g(\lambda) \leq \phi(b)$. So $\phi(b) = g(\lambda)$. So strong duality holds.

($\Rightarrow$). Assume now that we have strong duality. The there exists λ such that for all $c \in \mathbb{R}^m$,

$$\begin{aligned}\phi(b) &= g(\lambda) \\ &= \inf_{x\in X} L(x,\lambda) \\ &\leq \inf_{x\in X(c)} L(x,\lambda) \\ &= \inf_{x\in X(c)}(f(x) - \lambda^T(h(x)-b)) \\ &= \phi(c) - \lambda^T(c-b)\end{aligned}$$

So $\phi(b) + \lambda^T(c-b) \leq \phi(c)$. So this defines a supporting hyperplane. □

We are having some progress now. To show that Lagrange multipliers work, we need to show that (P) satisfies strong duality. To show that (P) satisfies strong duality, we need to show that it has a supporting hyperplane at b. How can we show that there is a supporting hyperplane? A sufficient condition is convexity.

Theorem (Supporting hyperplane theorem)**.** Suppose that $\phi : \mathbb{R}^m \to \mathbb{R}$ is convex and $b \in \mathbb{R}^m$ lies in the interior of the set of points where ϕ is finite. Then there exists a supporting hyperplane to ϕ at b.

Proof follows rather straightforwardly from the definition of convexity, and is omitted.

This is some even better progress. However, the definition of ϕ is rather convoluted. How can we show that it is convex? We have the following helpful theorem:

Theorem. Let

$$\phi(b) = \inf_{x\in X}\{f(x) : h(x) \leq b\}$$

If X, f, h are convex, then so is ϕ (assuming feasibility and boundedness).

Proof. Consider $b_1, b_2 \in \mathbb{R}^m$ such that $\phi(b_1)$ and $\phi(b_2)$ are defined. Let $\delta \in [0,1]$ and define $b = \delta b_1 + (1-\delta)b_2$. We want to show that $\phi(b) \leq \delta\phi(b_1) + (1-\delta)\phi(b_2)$.

Consider $x_1 \in X(b_1)$, $x_2 \in X(b_2)$, and let $x = \delta x_1 + (1-\delta)x_2$. By convexity of X, $x \in X$.

By convexity of h,

$$\begin{aligned} h(x) &= h(\delta x_1 + (1-\delta)x_2) \\ &\leq \delta h(x_1) + (1-\delta)h(x_2) \\ &\leq \delta b_1 + (1-\delta)b_2 \\ &= b \end{aligned}$$

So $x \in X(b)$. Since $\phi(x)$ is an optimal solution, by convexity of f,

$$\begin{aligned} \phi(b) &\leq f(x) \\ &= f(\delta x_1 + (1-\delta)x_2) \\ &\leq \delta f(x_1) + (1-\delta)f(x_2) \end{aligned}$$

This holds for any $x_1 \in X(b_1)$ and $x_2 \in X(b_2)$. So by taking infimum of the right hand side,

$$\phi(b) \leq \delta\phi(b_1) + (1-\delta)\phi(b_2).$$

So ϕ is convex. □

$h(x) = b$ is equivalent to $h(x) \leq b$ and $-h(x) \leq -b$. So the result holds for problems with equality constraints if both h and $-h$ are convex, i.e. if $h(x)$ is linear.

So

Theorem. If a linear program is feasible and bounded, then it satisfies strong duality.

3 Solutions of linear programs

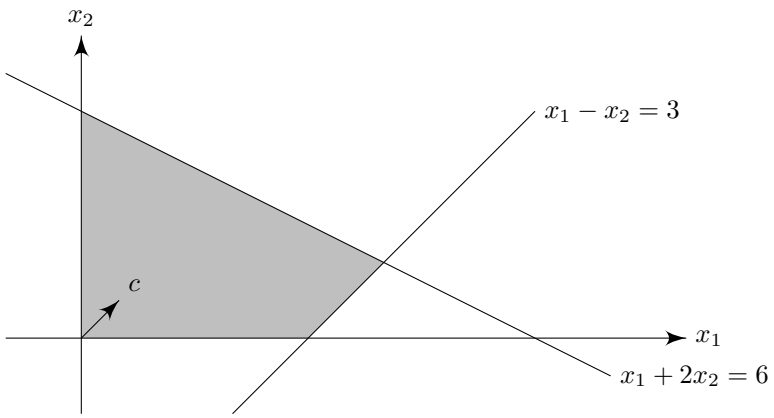
3.1 Linear programs

We'll come up with an algorithm to solve linear program efficiently. We first illustrate the general idea with the case of a 2D linear program. Consider the problem

$$\text{maximize } x_1 + x_2 \text{ subject to}$$

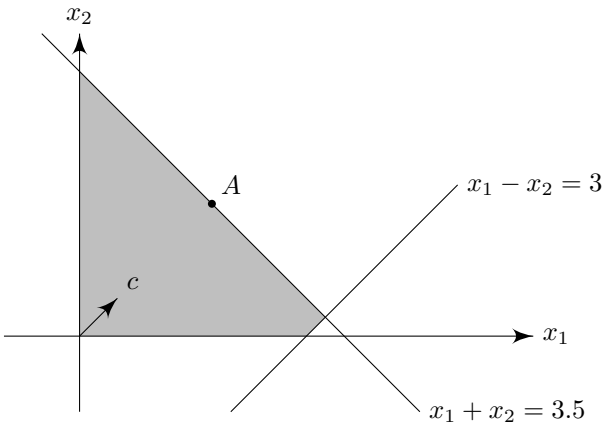
$$\begin{aligned} x_1 + 2x_2 &\leq 6 \\ x_1 - x_2 &\leq 3 \\ x_1, x_2 &\geq 0 \end{aligned}$$

We can plot the solution space out



To maximize $x_1 + x_2$, we want to go as far in the c direction as possible. It should be clear that the optimal point will lie on a corner of the polygon of feasible region, no matter what the shape of it might be.

Even if we have cases where c is orthogonal to one of the lines, eg



An optimal point might be A. However, if we know that A is an optimal point, we can slide it across the $x_1 + x_2 = 3.5$ line until it meets one of the corners. Hence we know that one of the corners must be an optimal point.

This already allows us to solve linear programs, since we can just try all corners and see which has the smallest value. However, this can be made more efficient, especially when we have a large number of dimensions and hence corners.

3.2 Basic solutions

Here we will assume that the rows of A are linearly independent, and any set of m columns are linearly independent. Otherwise, we can just throw away the redundant rows or columns.

In general, if both the constraints and the objective functions are linear, then the optimal point always lies on a "corner", or an *extreme point*.

Definition (Extreme point)**.** An *extreme point* $x \in S$ of a convex set S is a point that cannot be written as a convex combination of two distinct points in S, i.e. if $y, z \in S$ and $\delta \in (0, 1)$ satisfy

$$x = \delta y + (1 - \delta)z,$$

then $x = y = z$.

Consider again the linear program in standard form, i.e.

maximize $c^T x$ subject to $Ax = b, x \geq 0$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Note that now we are talking about maximization instead of minimization.

Definition (Basic solution and basis)**.** A solution $x \in \mathbb{R}^n$ is *basic* if it has at most m non-zero entries (out of n), i.e. if there exists a set $B \subseteq \{1, \cdots, n\}$ with $|B| = m$ such that $x_i = 0$ if $i \notin B$. In this case, B is called the *basis*, and x_i are the *basic variables* if $i \in B$.

We will later see (via an example) that basic solutions correspond to solutions at the "corners" of the solution space.

Definition (Non-degenerate solutions)**.** A basic solution is *non-degenerate* if it has exactly m non-zero entries.

Note that by "solution", we do not mean a solution to the whole maximization problem. Instead we are referring to a solution to the constraint $Ax = b$. Being a solution does *not* require that $x \geq 0$. Those that satisfy this regional constraint are known as *feasible*.

Definition (Basic feasible solution)**.** A basic solution x is *feasible* if it satisfies $x \geq 0$.

Example. Consider the linear program

$$\text{maximize } f(x) = x_1 + x_2 \text{ subject to}$$

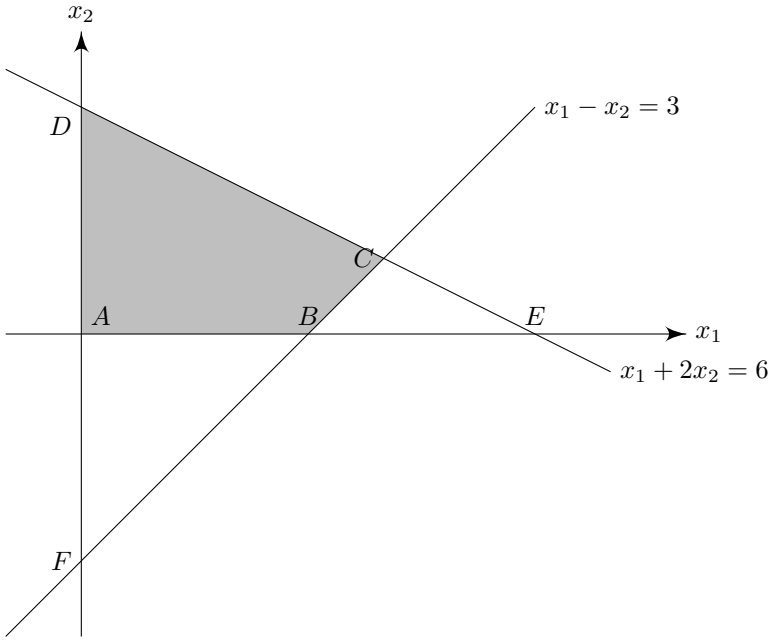
$$\begin{aligned} x_1 + 2x_2 + z_1 &= 6 \\ x_1 - x_2 + z_2 &= 3 \\ x_1, x_2, z_1, z_2 &\geq 0 \end{aligned}$$

where we have included the slack variables.

Since we have 2 constraints, a basic solution would require 2 non-zero entries, and thus 2 zero entries. The possible basic solutions are

	x_1	x_2	z_1	z_2	$f(x)$
A	0	0	6	3	0
B	0	3	0	6	3
C	4	1	0	0	5
D	3	0	3	0	3
E	6	0	0	-4	6
F	0	-3	12	0	-3

Among all 6, E and F are *not* feasible solutions since they have negative entries. So the basic feasible solutions are A, B, C, D.



In previous example, we saw that the extreme points are exactly the basic feasible solutions. This is true in general.

Theorem. A vector x is a basic feasible solution of $Ax = b$ if and only if it is an extreme point of the set $X(b) = \{x' : Ax' = b, x' \geq 0\}$.

We will not prove this.

3.3 Extreme points and optimal solutions

Recall that we previously showed in our 2D example that the optimal solution lies on an extreme point, i.e. is a basic feasible solution. This is also true in general.

Theorem. If (P) is feasible and bounded, then there exists an optimal solution that is a basic feasible solution.

Proof. Let x be optimal of (P). If x has at most non-zero entries, it is a basic feasible solution, and we are done.

Now suppose x has $r > m$ non-zero entries. Since it is not an extreme point, we have $y \neq z \in X(b)$, $\delta \in (0, 1)$ such that

$$x = \delta y + (1 - \delta)z.$$

We will show there exists an optimal solution strictly fewer than r non-zero entries. Then the result follows by induction.

By optimality of x, we have $c^T x \geq c^T y$ and $c^T x \geq c^T z$.

Since $c^T x = \delta c^T y + (1 - \delta)c^T z$, we must have that $c^T x = c^T y = c^T z$, i.e. y and z are also optimal.

Since $y \geq 0$ and $z \geq 0$, $x = \delta y + (1 - \delta)z$ implies that $y_i = z_i = 0$ whenever $x_i = 0$.

So the non-zero entries of y and z is a subset of the non-zero entries of x. So y and z have at most r non-zero entries, which must occur in rows where x is also non-zero.

If y or z has strictly fewer than r non-zero entries, then we are done. Otherwise, for any $\hat{\delta}$ (not necessarily in $(0, 1)$), let

$$x_{\hat{\delta}} = \hat{\delta} y + (1 - \hat{\delta})z = z + \hat{\delta}(y - z).$$

Observe that $x_{\hat{\delta}}$ is optimal for every $\hat{\delta} \in \mathbb{R}$.

Moreover, $y - z \neq 0$, and all non-zero entries of $y - z$ occur in rows where x is non-zero as well. We can thus choose $\hat{\delta} \in \mathbb{R}$ such that $x_{\hat{\delta}} \geq 0$ and $x_{\hat{\delta}}$ has strictly fewer than r non-zero entries. □

Intuitively, this is what we do when we "slide along the line" if c is orthogonal to one of the boundary lines.

This result in fact holds more generally for the maximum of a convex function f over a compact (i.e. closed and bounded) convex set X.

In that case, we can write any point $x \in X$ as a convex combination

$$x = \sum_{i=1}^{k} \delta_i x^i$$

of extreme points $x^k \in X$, where $\delta \in \mathbb{R}^k_{\geq 0}$ and $\sum_{i=1}^k \delta_i = 1$.

Then, by convexity of f,

$$f(x) \leq \sum_{i=1}^{k} \delta_i f(x^i) \leq \max_i f(x^i)$$

So any point in the interior cannot be better than the extreme points.

3.4 Linear programming duality

Consider the linear program in general form with slack variables,

$$\text{minimize } c^T x \text{ subject to } Ax - z = b,\ x, z \geq 0$$

We have $X = \{(x, z) : x, z \geq 0\} \subseteq \mathbb{R}^{m+n}$.

The Lagrangian is

$$L(x, z, \lambda) = c^T x - \lambda^T (Ax - z - b) = (c^T - \lambda^T A)x + \lambda^T z + \lambda^T b.$$

Since x, z can be arbitrarily positive, this has a finite minimum if and only if

$$c^T - \lambda^T A \geq 0, \quad \lambda^T \geq 0.$$

Call the feasible set Y. Then for fixed $\lambda \in Y$, the minimum of $L(x, z, \lambda)$ is attained when $(c^T - \lambda^T A)x$ and $\lambda^T z = 0$ by complementary slackness. So

$$g(\lambda) = \inf_{(x,z)\in X} L(x, z, \lambda) = \lambda^T b.$$

The dual is thus

$$\text{maximize } \lambda^T b \text{ subject to } A^T\lambda \leq c,\ \lambda \geq 0$$

Theorem. The dual of the dual of a linear program is the primal.

Proof. It suffices to show this for the linear program in general form. We have shown above that the dual problem is

$$\text{minimize } -b^T\lambda \text{ subject to } -A^T\lambda \geq -c,\ \lambda \geq 0.$$

This problem has the same form as the primal, with $-b$ taking the role of c, $-c$ taking the role of b, $-A^T$ taking the role of A. So doing it again, we get back to the original problem. □

Example. Let the primal problem be

$$\text{maximize } 3x_1 + 2x_2 \text{ subject to}$$

$$\begin{aligned} 2x_1 + x_2 + z_1 &= 4 \\ 2x_1 + 3x_2 + z_2 &= 6 \\ x_1, x_2, z_1, z_2 &\geq 0. \end{aligned}$$

Then the dual problem is

$$\text{minimize } 4\lambda_1 + 6\lambda_2 \text{ such that}$$

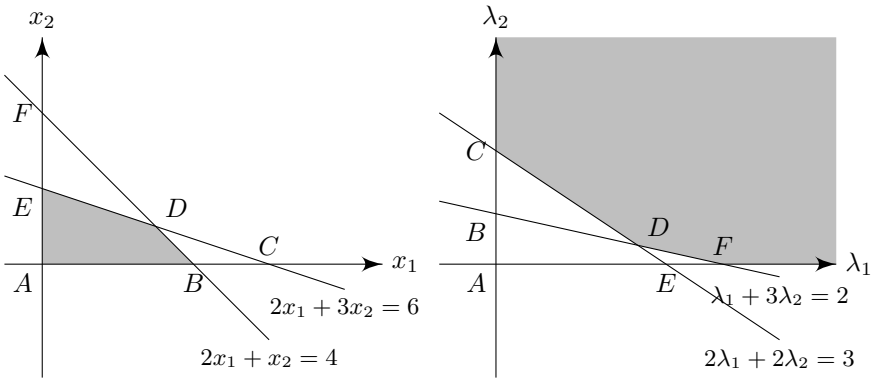
$$\begin{aligned} 2\lambda_1 + 2\lambda_2 - \mu_1 &= 3 \\ \lambda_1 + 3\lambda_2 - \mu_2 &= 2 \\ \lambda_1, \lambda_2, \mu_1, \mu_2 &\geq 0. \end{aligned}$$

We can compute all basic solutions of the primal and the dual by setting $n-m-2$ variables to be zero in turn.

Given a particular basic solutions of the primal, the corresponding solutions of the dual can be found by using the complementary slackness solutions:

$$\lambda_1 z_1 = \lambda_2 z_2 = 0, \quad \mu_1 x_1 = \mu_2 x_2 = 0.$$

	x_1	x_2	z_1	z_2	$f(x)$	λ_1	λ_2	μ_1	μ_2	$g(\lambda)$
A	0	0	4	6	0	0	0	-3	-2	0
B	2	0	0	2	6	$\frac{3}{2}$	0	0	$-\frac{1}{2}$	6
C	3	0	-2	0	9	0	$\frac{3}{2}$	0	$\frac{5}{2}$	9
D	$\frac{3}{2}$	1	0	0	$\frac{13}{2}$	$\frac{5}{4}$	$\frac{1}{4}$	0	0	$\frac{13}{2}$
E	0	2	2	0	4	0	$\frac{2}{3}$	$-\frac{5}{3}$	0	4
F	0	4	0	-6	8	2	0	1	0	8



We see that D is the only solution such that both the primal and dual solutions are feasible. So we know it is optimal without even having to calculate $f(x)$. It turns out this is always the case.

Theorem. Let x and λ be feasible for the primal and the dual of the linear program in general form. Then x and λ and optimal if and only if they satisfy complementary slackness, i.e. if

$$(c^T - \lambda^T A)x = 0 \text{ and } \lambda^T(Ax - b) = 0.$$

Proof. If x and λ are optimal, then

$$c^T x = \lambda^T b$$

since every linear program satisfies strong duality. So

$$\begin{aligned} c^T x &= \lambda^T b \\ &= \inf_{x' \in X}(c^T x' - \lambda^T(Ax' - b)) \\ &\leq c^T x - \lambda^T(Ax - b) \\ &\leq c^T x. \end{aligned}$$

The last line is since $Ax \geq b$ and $\lambda \geq 0$.

The first and last term are the same. So the inequalities hold with equality. Therefore

$$\lambda^T b = c^T x - \lambda^T(Ax - b) = (c^T - \lambda^T A)x + \lambda^T b.$$

So

$$(c^T - \lambda^T A)x = 0.$$

Also,

$$c^T x - \lambda^T(Ax - b) = c^T x$$

implies

$$\lambda^T(Ax - b) = 0.$$

On the other hand, suppose we have complementary slackness, i.e.

$$(c^T - \lambda^T A)x = 0 \text{ and } \lambda^T(Ax - b) = 0,$$

then

$$c^T x = c^T x - \lambda^T(Ax - b) = (c^T - \lambda^T A)x + \lambda^T b = \lambda^T b.$$

Hence by weak duality, x and λ are optimal. □

3.5 Simplex method

The simplex method is an algorithm that makes use of the result we just had. To find the optimal solution to a linear program, we start with a basic feasible solution of the primal, and then modify the variables step by step until the dual is also feasible.

We start with an example, showing what we do, then explain the logic behind, then do a more proper example.

Example. Consider the following problem:

$$\text{maximize } x_1 + x_2 \text{ subject to}$$

$$\begin{aligned} x_1 + 2x_2 + z_1 &= 6 \\ x_1 - x_2 + z_2 &= 3 \\ x_1, x_2, z_1, z_2 &\geq 0. \end{aligned}$$

We write everything in the *simplex tableau*, by noting down the coefficients:

	x_1	x_2	z_1	z_2	
Constraint 1	1	2	1	0	6
Constraint 2	1	-1	0	1	3
Objective	1	1	0	0	0

We see an identity matrix $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ in the z_1 and z_2 columns, and these correspond to basic feasible solution: $z_1 = 6, z_2 = 3, x_1 = x_2 = 0$. It's pretty clear that our basic feasible solution is not optimal, since our objective function is 0. This is since something in the last row is positive, and we can increase the objective by, say, increasing x_1.

The simplex method says that we can find the optimal solution if we make the bottom row all negative while keeping the right column positive, by doing row operations.

We multiply the first row by $\frac{1}{2}$ and subtract/add it to the other rows to obtain

	x_1	x_2	z_1	z_2	
Constraint 1	$\frac{1}{2}$	1	$\frac{1}{2}$	0	3
Constraint 2	$\frac{2}{3}$	0	$\frac{1}{2}$	1	6
Objective	$\frac{1}{2}$	0	$-\frac{1}{2}$	0	-3

Our new basic feasible solution is $x_2 = 3, z_2 = 6, x_1 = z_1 = 0$. We see that the number in the bottom-right corner is $-f(x)$. We can continue this process to finally obtain a solution.

Here we adopt the following notation: let $A \subseteq \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Assume that A has full rank. Let B be a basis and set $B \subseteq \{1, 2, \cdots, n\}$ with $|B| = m$, corresponding to at most m non-zero entries.

We rearrange the columns so that all basis columns are on the left. Then we can write our matrices as

$$\begin{aligned} A_{m\times n} &= \begin{pmatrix}(A_B)_{m\times m} & (A_N)_{m\times(n-m)}\end{pmatrix} \\ x_{n\times 1} &= \begin{pmatrix}(x_B)_{m\times 1} & (x_N)_{(n-m)\times 1}\end{pmatrix}^T \\ c_{1\times n} &= \begin{pmatrix}(c_B)_{m\times 1} & (c_N)_{(n-m)\times 1}\end{pmatrix}. \end{aligned}$$

Then the functional constraints

$$Ax = b$$

can be decomposed as

$$A_B x_B + A_N x_N = b.$$

We can rearrange this to obtain

$$x_B = A_B^{-1}(b - A_N x_N).$$

In particular, when $x_N = 0$, then

$$x_B = A_B^{-1} b.$$

The general tableau is then

Basis components	Other components	
$A_B^{-1}A_B = I$	$A_B^{-1}A_N$	$A_B^{-1}b$
$c_B^T - c_B^T A_B^{-1} A_B = 0$	$c_N^T - c_B^T A_B^{-1} A_N$	$-c_B^T A_B^{-1} b$

This might look really scary, and it is! Without caring too much about how the formulas for the cells come from, we see the identity matrix on the left, which is where we find our basic feasible solution. Below that is the row for the objective function. The values of this row must be 0 for the basis columns.

On the right-most column, we have $A_B^{-1}b$, which is our x_B. Below that is $-c_B^T A_B^{-1} b$, which is the negative of our objective function $c_B^T x_B$.

3.5.1 The simplex tableau

We have

$$\begin{aligned} f(x) &= c^T x \\ &= c_B^T x_B + c_N^T x_N \\ &= c_B^T A_B^{-1}(b - A_N x_N) + c_N^T x_N \\ &= c_B^T A_B^{-1} b + (c_N^T - c_B^T A_B^{-1} A_N) x_N. \end{aligned}$$

We will maximize $c^T x$ by choosing a basis such that $c_N^T - c_B^T A_B^{-1} A_N \leq 0$, i.e. non-positive everywhere and $A_B^{-1} b \geq 0$.

If this is true, then for any feasible solution $x \in \mathbb{R}^n$, we must have $x_N \geq 0$. So $(c_N^T - c_B^T A_B^{-1} A_N) x_N \leq 0$ and

$$f(x) \leq c_B^T A_B^{-1} b.$$

So if we choose $x_B = A_B^{-1}b$, $x_N = 0$, then we have an optimal solution.

Hence our objective is to pick a basis that makes $c_N^T - c_B^T A_B^{-1} A_N \leq 0$ while keeping $A_B^{-1}b \geq 0$. To do this, suppose this is not attained. Say $(c_N^T - c_B^T A_B^{-1} A_N)_i > 0$.

We can increase the value of the objective function by increasing $(x_N)_i$. As we increase $(x_N)_i$, we have to satisfy the functional constraints. So the value of other variables will change as well. We can keep increasing $(x_N)_i$ until another variable hits 0, say $(x_B)_j$. Then we will have to stop.

(However, if it so happens that we can increase $(x_N)_i$ indefinitely without other things hitting 0, our problem is unbounded)

The effect of this is that we have switched basis by removing $(x_B)_j$ and adding $(x_N)_i$. We can continue from here. If $(c_N^T - c_B^T A_B^{-1} A_N)$ is negative, we are done. Otherwise, we continue the above procedure.

The simplex method is a systematic way of doing the above procedure.

3.5.2 Using the Tableau

Consider a tableau of the form

a_{ij}	a_{i0}
a_{0j}	a_{00}

where a_{i0} is b, a_{0j} corresponds to the objective function, and a_{00} is initial 0.

The simplex method proceeds as follows:

(i) Find an initial basic feasible solution.

(ii) Check whether $a_{0j} \leq 0$ for every j. If so, the current solution is optimal. Stop.

(iii) If not, choose a *pivot column* j such that $a_{0j} > 0$. Choose a *pivot row* $i \in \{i : a_{ij} > 0\}$ that minimizes a_{i0}/a_{ij}. If multiple rows are minimize a_{i0}/a_{ij}, then the problem is degenerate, and things *might* go wrong. If $a_{ij} \leq 0$ for all i, i.e. we cannot choose a pivot row, the problem is unbounded, and we stop.

(iv) We update the tableau by multiplying row i by $1/a_{ij}$ (such that the new $a_{ij} = 1$), and add a $(-a_{kj}/a_{ij})$ multiple of row i to each row $k \neq i$, including $k = 0$ (so that $a_{kj} = 0$ for all $k \neq i$)

We have a basic feasible solution, since our choice of a_{ij} makes all right-hand columns positive after subtracting (apart from a_{00}).

(v) `GOTO` (ii).

Now visit the example at the beginning of the section to see how this is done in practice. Then read the next section for a more complicated example.

3.6 The two-phase simplex method

Sometimes we don't have a nice identity matrix to start with. In this case, we need to use the *two-phase simplex method* to first find our first basic feasible solution, then to the actual optimization.

This method is illustrated by example.

Example. Consider the problem

$$\text{minimize } 6x_1 + 3x_2 \text{ subject to}$$

$$\begin{aligned} x_1 + x_2 &\geq 1\\ 2x_1 - x_2 &\geq 1\\ 3x_2 &\leq 2\\ x_1, x_2 &\geq 0 \end{aligned}$$

This is a minimization problem. To avoid being confused, we maximize $-6x_1-3x_2$ instead. We add slack variables to obtain

$$\text{maximize } -6x_1 - 3x_2 \text{ subject to}$$

$$\begin{aligned} x_1 + x_2 - z_1 &= 1\\ 2x_1 - x_2 - z_2 &= 1\\ 3x_2 + z_3 &= 2\\ x_1, x_2, z_1, z_2, z_3 &\geq 0 \end{aligned}$$

Now we don't have a basic feasible solution, since we would need $z_1 = z_2 = -1, z_3 = 2$, which is not feasible. So we add *more* variables, called the artificial variables.

$$\text{maximize } -6x_1 - 3x_2 \text{ subject to}$$

$$\begin{aligned} x_1 + x_2 - z_1 + y_1 &= 1\\ 2x_1 - x_2 - z_2 + y_2 &= 1\\ 3x_2 + z_3 &= 2\\ x_1, x_2, z_1, z_2, z_3, y_1, y_2 &\geq 0 \end{aligned}$$

Note that adding y_1 and y_2 might create new solutions, which is bad. We solve this problem by first trying to make y_1 and y_2 both 0 and find a basic feasible solution. Then we can throw away y_1 and y_2 and then get a basic feasible for our original problem. So momentarily, we want to solve

$$\text{minimize } y_1 + y_2 \text{ subject to}$$

$$\begin{aligned} x_1 + x_2 - z_1 + y_1 &= 1\\ 2x_1 - x_2 - z_2 + y_2 &= 1\\ 3x_2 + z_3 &= 2\\ x_1, x_2, z_1, z_2, z_3, y_1, y_2 &\geq 0 \end{aligned}$$

By minimizing y_1 and y_2, we will make them zero.

Our simplex tableau is

x_1	x_2	z_1	z_2	z_3	y_1	y_2	
1	1	-1	0	0	1	0	1
2	-1	0	-1	0	0	1	1
0	3	0	0	1	0	0	2
-6	-3	0	0	0	0	0	0
0	0	0	0	0	-1	-1	0

Note that we keep both our original and "kill-y_i" objectives, but now we only care about the second one. We will keep track of the original objective so that we can use it in the second phase.

We see an initial feasible solution $y_1 = y_2 = 1, z_3 = 2$. However, this is not a proper simplex tableau, as the basis columns should not have non-zero entries (apart from the identity matrix itself). But we have the two -1s at the bottom! So we add the first two rows to the last to obtain

x_1	x_2	z_1	z_2	z_3	y_1	y_2	
1	1	-1	0	0	1	0	1
2	-1	0	-1	0	0	1	1
0	3	0	0	1	0	0	2
-6	-3	0	0	0	0	0	0
3	0	-1	-1	0	0	0	2

Our pivot column is x_1, and our pivot row is the second row. We divide it by 1 and add/subtract it from other rows.

x_1	x_2	z_1	z_2	z_3	y_1	y_2	
0	$\frac{3}{2}$	-1	$\frac{1}{2}$	0	1	$-\frac{1}{2}$	$\frac{1}{2}$
1	$-\frac{1}{2}$	0	$-\frac{1}{2}$	0	0	$\frac{1}{2}$	$\frac{1}{2}$
0	3	0	0	1	0	0	2
0	-6	0	-3	0	0	3	3
0	$\frac{3}{2}$	-1	$\frac{1}{2}$	0	0	$-\frac{3}{2}$	$\frac{1}{2}$

There are two possible pivot columns. We pick z_2 and use the first row as the pivot row.

x_1	x_2	z_1	z_2	z_3	y_1	y_2	
0	3	-2	1	0	2	-1	1
1	1	-1	0	0	1	0	1
0	3	0	0	1	0	0	2
0	3	-6	0	0	6	0	6
0	0	0	0	0	-1	-1	0

We see that y_1 and y_2 are no longer in the basis, and hence take value 0. So we drop all the phase I stuff, and are left with

x_1	x_2	z_1	z_2	z_3	
0	3	-2	1	0	1
1	1	-1	0	0	1
0	3	0	0	1	2
0	3	-6	0	0	6

We see a basic feasible solution $z_1 = x_1 = 1, z_3 = 2$.

We pick x_2 as the pivot column, and the first row as the pivot row. Then we have

x_1	x_2	z_1	z_2	z_3	
0	1	$-\frac{2}{3}$	$\frac{1}{3}$	0	$\frac{1}{3}$
1	0	$-\frac{1}{3}$	$-\frac{1}{3}$	0	$\frac{2}{3}$
0	0	2	-1	1	1
0	0	-4	-1	0	5

Since the last row is all negative, we have complementary slackness. So this is a optimal solution. So $x_1 = \frac{2}{3}, x_2 = \frac{1}{3}, z_3 = 1$ is a feasible solution, and our optimal value is 5.

Note that we previously said that the bottom right entry is the negative of the optimal value, not the optimal value itself! This is correct, since in the tableau, we are maximizing $-6x_1 - 3x_2$, whose maximum value is -5. So the minimum value of $6x_1 + 3x_2$ is 5.

4 Non-cooperative games

Here we have a short digression to game theory. We mostly focus on games with two players.

4.1 Games and Solutions

Definition (Bimatrix game)**.** A two-player game, or *bimatrix game*, is given by two matrices $P, Q \in \mathbb{R}^{m\times n}$. Player 1, or the *row player*, chooses a row $i \in \{1, \cdots, m\}$, while player 2, the *column player*, chooses a column $j \in \{1, \cdots, n\}$. These are selected without knowledge of the other player's decisions. The two players then get payoffs P_{ij} and Q_{ij} respectively.

Example. A game of rock-paper-scissors can have payoff matrices

$$P_{ij} = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}, \quad Q_{ij} = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix}.$$

Here a victory gives you a payoff of 1, a loss gives a payoff of -1, and a draw gives a payoff of 1. Also the first row/column corresponds to playing rock, second corresponds to paper and third corresponds to scissors.

Usually, this is not the best way to display the payoff matrices. First of all, we need two write out two matrices, and there isn't an easy way to indicate what row corresponds to what decision. Instead, we usually write this as a table.

	R	P	S
R	$(0, 0)$	$(-1, 1)$	$(1, -1)$
P	$(1, -1)$	$(0, 0)$	$(-1, 1)$
S	$(-1, 1)$	$(1, -1)$	$(0, 0)$

By convention, the first item in the tuple $(-1, 1)$ indicates the payoff of the row player, and the second item indicates the payoff of the column player.

Definition (Strategy)**.** Players are allowed to play randomly. The set of *strategies* the row player can have is

$$X = \{x \in \mathbb{R}^m : x \geq 0, \sum x_i = 1\}$$

and the column player has strategies

$$Y = \{y \in \mathbb{R}^n : y \geq 0, \sum y_i = 1\}$$

Each vector corresponds to the probabilities of selecting each row or column.

A strategy profile $(x, y) \in X \times Y$ induces a lottery, and we write $p(x, y) = x^T P y$ for the expected payoff of the row player.

If $x_i = 1$ for some i, i.e. we always pick i, we call x a *pure strategy*.

Example (Prisoner's dilemma)**.** Suppose Alice and Bob commit a crime together, and are caught by the police. They can choose to remain silent (S) or testify (T). Different options will lead to different outcomes:

– Both keep silent: the police has little evidence and they go to jail for 2 years.

– One testifies and one remains silent: the one who testifies gets awarded and is freed, while the other gets stuck in jail for 10 years.

– Both testify: they both go to jail for 5 years.

We can represent this by a payoff table:

	S	T
S	$(2,2)$	$(0,3)$
T	$(3,0)$	$(1,1)$

Note that higher payoff is desired, so a longer serving time corresponds to a lower payoff. Also, payoffs are interpreted relatively, so replacing $(0,3)$ with $(0,100)$ (and $(3,0)$ with $(100,0)$) in the payoff table would make no difference.

Here we see that regardless of what the other person does, it is always strictly better to testify than not (unless you want to be nice). We say T is a *dominant strategy*, and $(1,1)$ is *Pareto dominated* by $(2,2)$.

Example (Chicken)**.** The game of *Chicken* is as follows: two people drive their cars towards each other at high speed. If they collide, they will di.e. Hence they can decide to chicken out (C) or continue driving (D). If both don't chicken, they die, which is bad. If one chickens and the other doesn't the person who chicken looks silly (but doesn't die). If both chicken out, they both look slightly silly. This can be represented by the following table:

	C	D
C	$(2,2)$	$(1,3)$
D	$(3,1)$	$(0,0)$

Here there is no dominating strategy, so we need a different way of deciding what to do.

Instead, we define the *security level* of the row player to be

$$\max_{x\in X}\min_{y\in Y} p(x,y) = \max_{x\in X}\min_{j\in\{1,\dots,n\}}\sum_{i=1}^{m} x_i p_{ij}.$$

Such an x is the strategy the row player can employ that minimizes the worst possible loss. This is called the maximin strategy.

We can formulate this as a linear program:

$$\text{maximize } v \text{ such that}$$

$$\sum_{i=1}^{m} x_i p_{ij} \geq v \quad \text{for all } j = 1, \cdots, n$$

$$\sum_{i=1}^{m} x_i = 1$$

$$x \geq 0$$

Here the maximin strategy is to chicken. However, this isn't really what we are looking for, since if both players employ this maximin strategy, it would be better for you to not chicken out.

Definition (Best response and equilibrium)**.** A strategy $x \in X$ is a *best response* to $y \in Y$ if for all $x' \in X$

$$p(x, y) \geq p(x', y)$$

A pair (x, y) is an *equilibrium* if x is the best response against y and y is a best response against x.

Example. In the chicken game, there are two pure equilibrium, $(3, 1)$ and $(1, 3)$, and there is a mixed equilibrium in which the players pick the options with equal probability.

Theorem (Nash, 1961)**.** Every bimatrix game has an equilibrium.

We are not proving this since it is too hard.

4.2 The minimax theorem

There is a special type of game known as a *zero sum game*.

Definition (Zero-sum game)**.** A bimatrix game is a *zero-sum game*, or matrix game, if $q_{ij} = -p_{ij}$ for all i, j, i.e. the total payoff is always 0.

To specify a matrix game, we only need one matrix, not two, since the matrix of the other player is simply the negative of the matrix of the first.

Example. The rock-paper-scissors games as specified in the beginning example is a zero-sum game.

Theorem (von Neumann, 1928)**.** If $P \in \mathbb{R}^{m \times n}$. Then

$$\max_{x \in X} \min_{y \in Y} p(x, y) = \min_{y \in Y} \max_{x \in X} p(x, y).$$

Note that this is equivalent to

$$\max_{x \in X} \min_{y \in Y} p(x, y) = -\max_{y \in Y} \min_{x \in X} -p(x, y).$$

The left hand side is the worst payoff the row player can get if he employs the minimax strategy. The right hand side is the worst payoff the column player can get if he uses his minimax strategy.

The theorem then says that if both players employ the minimax strategy, then this is an equilibrium.

Proof. Recall that the optimal value of $\max \min p(x, y)$ is a solution to the linear program

$$\begin{aligned} &\text{maximize } v \text{ such that} \\ &\sum_{i=1}^{m} x_i p_{ij} \geq v \quad \text{for all } j = 1, \cdots, n \\ &\sum_{i=1}^{m} x_i = 1 \\ &x \geq 0 \end{aligned}$$

Adding slack variable $z \in \mathbb{R}^n$ with $z \geq 0$, we obtain the Lagrangian

$$L(v, x, z, w, y) = v + \sum_{j=1}^{n} y_j \left(\sum_{i=1}^{m} x_i p_{ij} - z_j - v \right) - w \left(\sum_{i=1}^{m} x_i - 1 \right),$$

where $w \in \mathbb{R}$ and $y \in \mathbb{R}^n$ are Lagrange multipliers. This is equal to

$$\left(1 - \sum_{j=1}^{n} y_j \right) v + \sum_{i=1}^{m} \left(\sum_{j=1}^{n} p_{ij} y_j - w \right) x_i - \sum_{j=1}^{n} y_j z_j + w.$$

This has finite minimum for all $v \in \mathbb{R}, x \geq 0$ iff $\sum y_i = 1$, $\sum p_{ij} y_j \leq w$ for all i, and $y \geq 0$. The dual is therefore

$$\text{minimize } w \text{ subject to}$$

$$\sum_{j=1}^{n} p_{ij} y_j \leq w \quad \text{for all } i$$

$$\sum_{j=1}^{n} y_j = 1$$

$$y \geq 0$$

This corresponds to the column player choosing a strategy (y_i) such that the expected payoff is bounded above by w.

The optimum value of the dual is $\min_{y \in Y} \max_{x \in X} p(x, y)$. So the result follows from strong duality. □

Definition (Value)**.** The *value* of the matrix game with payoff matrix P is

$$v = \max_{x \in X} \min_{y \in Y} p(x, y) = \min_{y \in Y} \max_{x \in X} p(x, y).$$

In general, the equilibrium are given by

Theorem. $(x, y) \in X \times Y$ is an equilibrium of the matrix game with payoff matrix P if and only if

$$\min_{y' \in Y} p(x, y') = \max_{x' \in X} \min_{y' \in Y} p(x', y')$$
$$\max_{x' \in X} p(x', y) = \min_{y' \in Y} \max_{x' \in X} p(x', u')$$

i.e. the x, y are optimizers for the max min and min max functions.

Proof is in the second example sheet.

5 Network problems

5.1 Definitions

We are going to look into several problems that involve graphs. Unsurprisingly, we will need some definitions from graph theory.

Definition (Directed graph/network)**.** A *directed graph* or *network* is a pair $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. If $(u, v) \in E$, we say there is an edge from u to v.

Definition (Degree)**.** The degree of a vertex $u \in V$ is the number of $v \in V$ such that $(u, v) \in E$ or $(v, u) \in E$.

Definition (Walk)**.** An *walk* from $u \in V$ to $v \in V$ is a sequence of vertices $u = v_1, \cdots, v_k = v$ such that $(v_i, v_{i+1}) \in E$ for all i. An *undirected walk* allows $(v_i, v_{i+1}) \in E$ or $(v_{i+1}, v) \in E$, i.e. we are allowed to walk backwards.

Definition (Path)**.** A path is a walk where $v_1, \cdots, v_k$ are pairwise distinct.

Definition (Cycle)**.** A cycle is a walk where $v_1, \cdots, v_{k-1}$ are pairwise distinct and $v_1 = v_k$.

Definition (Connected graph)**.** A graph is *connected* if for any pair of vertices, there is an undirected path between them.

Definition (Tree)**.** A *tree* is a connected graph without (undirected) cycles.

Definition (Spanning tree)**.** The *spanning tree* of a graph $G = (V, E)$ is a tree (V', E') with $V' = V$ and $E' \subseteq E$.

5.2 Minimum-cost flow problem

Let $G = (V, E)$ be a directed graph. Let the number of vertices be $|V| = n$ and let $b \in \mathbb{R}^n$. For each edge, we assign three numbers: a cost, an lower bound and a upper bound. We denote these as matrices as $C, \underline{M}, \overline{M} \in \mathbb{R}^{n \times n}$.

Each component of the vector b_i denotes the amount of flow entering or leaving each vertex $i \in V$. If $b_i > 0$, we call $i \in V$ a source. For example, if we have a factory at b_i that produces stuff, b_i will be positive. This is only the amount of stuff produced or consumed in the vertex, and not how many things flow through the vertex.

c_{ij} is the cost of transferring one unit of stuff from vertex i to vertex j (fill entries with 0 if there is no edge between the vertices), and $\underline{m}_{ij}$ and $\overline{m}_{ij}$ denote the lower and upper bounds on the amount of flow along $(i, j) \in E$ respectively.

$x \in \mathbb{R}^{n \times n}$ is a minimum-cost flow if it minimizes the cost of transferring stuff, while satisfying the constraints, i.e. it is an optimal solution to the problem

$$\text{minimize} \sum_{(i,j) \in E} c_{ij} x_{ij} \text{ subject to}$$

$$b_i + \sum_{j:(j,i) \in E} x_{ji} = \sum_{j:(i,j) \in E} x_{ij} \quad \text{for each } i \in V$$

$$\underline{m}_{ij} \leq x_{ij} \leq \overline{m}_{ij} \quad \text{for all } (i, j) \in E.$$

This problem is a linear program. In theory, we can write it into the general form $Ax = b$, where A is a huge matrix given by

$$a_{ij} \begin{cases} 1 & \text{if the } k\text{th edge starts at vertex } i \\ -1 & \text{if the } k\text{th edge ends at vertex } i \\ 0 & \text{otherwise} \end{cases}$$

However, using this huge matrix to solve this problem by the simplex method is not very efficient. So we will look for better solutions.

Note that for the system to make sense, we must have

$$\sum_{i \in V} b_i = 0,$$

i.e. the total supply is equal to the total consumption.

To simplify the problem, we can convert it into an equivalent *circulation problem*, where $b_i = 0$ for all i. We do this by adding an additional vertex where we send all the extra b_i to. For example, if a vertex has $b_i = -1$, then it takes in more stuff than it gives out. So we can mandate it to send out one extra unit to the additional vertex. Then $b_i = 0$.

An *uncapacitated problem* is the case where $\underline{m}_{ij} = 0$ and $\overline{m}_{ij} = \infty$ for all $(i, j) \in E$. An uncapacitated problem is either unbounded or bounded. If it is bounded, then it is equivalent to a problem with finite capacities, since we can add a bound greater than what the optimal solution wants.

We are going to show that this can be reduced to a simpler problem:

5.3 The transportation problem

The transportation problem is a special case of the minimum-flow problem, where the graph is a *bipartite graph*. In other words, we can split the vertices into two halves A, B, where all edges flow from a vertex in A to a vertex in B. We call the vertices of A the *suppliers* and the vertices of B the *consumers*.

In this case, we can write the problem as

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \text{ subject to}$$

$$\sum_{j=1}^{m} x_{ij} = s_i \text{ for } i = 1, \cdots, n$$

$$\sum_{i=1}^{n} x_{ij} = d_j \text{ for } j = 1, \cdots, m$$

$$x \geq 0$$

This s_i is the supply of each supplier, and d_i is the demand of each supplier. We have $s \in \mathbb{R}^n, d \in \mathbb{R}^m$ satisfying $s, d \geq 0$, $\sum s_i = \sum d_j$.

Finally, we have $c \in \mathbb{R}^{n \times m}$ representing the cost of transferal.

We now show that every (bounded) minimum cost-flow problem can be reduced to the transportation problem.

Theorem. Every minimum cost-flow problem with finite capacities or non-negative costs has an equivalent transportation problem.

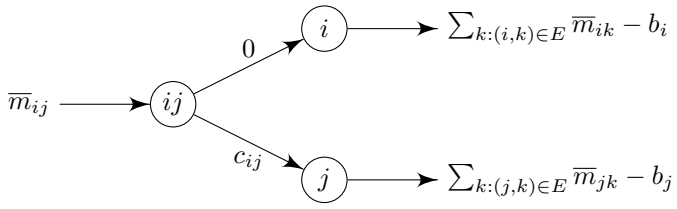
Proof. Consider a minimum-cost flow problem on network (V, E). It is wlog to assume that $\underline{m}_{ij} = 0$ for all $(i, j) \in E$. Otherwise, set $\underline{m}_{ij}$ to 0, $\overline{m}_{ij}$ to $\overline{m}_{ij} - \underline{m}_{ij}$, b_i to $b_i - \underline{m}_{ij}$, b_j to $b_j + \underline{m}_{ij}$, x_{ij} to $x_{ij} - \underline{m}_{ij}$. Intuitively, we just secretly ship the minimum amount without letting the network know.

Moreover, we can assume that all capacities are finite: if some edge has infinite capacity but non-negative cost, then setting the capacity to a large enough number, for example $\sum_{i \in V} |b_i|$ does not affect the optimal solutions. This is since cost is non-negative, and the optimal solution will not want shipping loops. So we will have at most $\sum |b_i|$ shipments.

We will construct an instance of the transportation problem as follows:

For every $i \in V$, add a consumer with demand $\left(\sum_{k:(i,k)\in E} \overline{m}_{ik}\right) - b_i$.

For every $(i, j) \in E$, add a supplier with supply $\overline{m}_{ij}$, an edge to consumer i with cost $c_{(ij,i)} = 0$ and an edge to consumer j with cost $c_{(ij,j)} = c_{ij}$.



The idea is that if the capacity of the edge (i, j) is, say, 5, in the original network, and we want to transport 3 along this edge, then in the new network, we send 3 units from ij to j, and 2 units to i.

The tricky part of the proof is to show that we have the same constraints in both graphs.

For any flow x in the original network, the corresponding flow on (ij, j) is x_{ij} and the flow on (ij, i) is $\overline{m}_{ij} - x_{ij}$. The total flow into i is then

$$\sum_{k:(i,k)\in E} (\overline{m}_{ik} - x_{ik}) + \sum_{k:(k,i)\in E} x_{ki}$$

This satisfies the constraints of the new network if and only if

$$\sum_{k:(i,k)\in E} (\overline{m}_{ik} - x_{ik}) + \sum_{k:(k,i)\in E} x_{ki} = \sum_{k:(i,k)\in E} \overline{m}_{ik} - b_i,$$

which is true if and only if

$$b_i + \sum_{k:(k,i)\in E} x_{ki} - \sum_{k:(i,k)\in E} x_{ik} = 0,$$

which is exactly the constraint for the node i in the original minimal-cost flow problem. So done. □

To solve the transportation problem, it is convenient to have two sets of Lagrange multipliers, one for the supplier constraints and one for the consumer

constraint. Then the Lagrangian of the transportation problem can be written as

$$L(x, \lambda, \mu) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{n}\lambda_i\left(s_i - \sum_{j=1}^{m} x_{ij}\right) - \sum_{j=1}^{n}\mu_j\left(d_j - \sum_{j=1}^{n} x_{ij}\right).$$

Note that we use different signs for the Lagrange multipliers for the suppliers and the consumers, so that our ultimate optimality condition will look nicer.

This is equivalent to

$$L(x, \lambda, \mu) = \sum_{i=1}^{n}\sum_{j=1}^{n}(c_{ij} - \lambda_i + \mu_j)x_{ij} + \sum_{i=1}^{n}\lambda_i s_i - \sum_{j=1}^{m}\mu_j d_j.$$

Since $x \geq 0$, the Lagrangian has a finite minimum iff $c_{ij} - \lambda_i + \mu_j \geq 0$ for all i, j. So this is our dual feasibility condition.

At an optimum, complementary slackness entails that

$$(c_{ij} - \lambda_i + \mu_j)x_{ij} = 0$$

for all i, j.

In this case, we have a tableau as follows:

	μ_1		μ_2		μ_3		μ_4		
	$\lambda_1 - \mu_1$		$\lambda_1 - \mu_2$		$\lambda_1 - \mu_3$		$\lambda_1 - \mu_4$		
λ_1	x_{11}	c_{11}	x_{12}	c_{12}	x_{13}	c_{13}	x_{14}	c_{14}	s_1
	$\lambda_2 - \mu_1$		$\lambda_2 - \mu_2$		$\lambda_2 - \mu_3$		$\lambda_2 - \mu_4$		
λ_2	x_{21}	c_{21}	x_{22}	c_{22}	x_{23}	c_{23}	x_{24}	c_{24}	s_1
	$\lambda_3 - \mu_1$		$\lambda_3 - \mu_2$		$\lambda_3 - \mu_3$		$\lambda_3 - \mu_4$		
λ_3	x_{31}	c_{31}	x_{32}	c_{32}	x_{33}	c_{33}	x_{34}	c_{34}	s_1
	d_1		d_2		d_3		d_4		

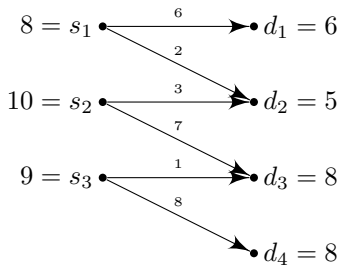
We have a row for each supplier and a column for each consumer.

Example. Suppose we have three suppliers with supplies $8, 10$ and 9; and four consumers with demands $6, 5, 8, 8$.

It is easy to create an initial feasible solution - we just start from the first consumer and first supplier, and supply as much as we can until one side runs out of stuff.

We first fill our tableau with our feasible solution.

6	5	2	3		4		6	8
	2	3	7	7	4		1	10
	5		6	1	2	8	4	9
6		5		8		8		



We see that our basic feasible solution corresponds to a spanning tree. In general, if we have n suppliers and m consumers, then we have $n+m$ vertices, and hence $n+m-1$ edges. So we have $n+m-1$ dual constraints. So we can arbitrarily choose one Lagrange multiplier, and the other Lagrange multipliers will follow. We choose $\lambda_1 = 0$. Since we require

$$(c_{ij} - \lambda_i + \mu_i)x_{ij} = 0,$$

for edges in the spanning tree, $x_{ij} \neq 0$. So $c_{ij} - \lambda_i + \mu_i = 0$. Hence we must have $\mu_1 = -5$. We can fill in the values of the other Lagrange multipliers as follows, and obtain

	-5	-3	0	-2
0	6 [5]	2 [3]	[4]	[6]
4	[2]	3 [7]	7 [4]	[1]
2	[5]	[6]	1 [2]	8 [4]

We can fill in the values of $\lambda_i - \mu_i$:

	-5	-3	0	-2
			0	2
0	6 [5]	2 [3]	[4]	[6]
	9			6
4	[2]	3 [7]	7 [4]	[1]
	7	5		
2	[5]	[6]	1 [2]	8 [4]

The dual feasibility condition is

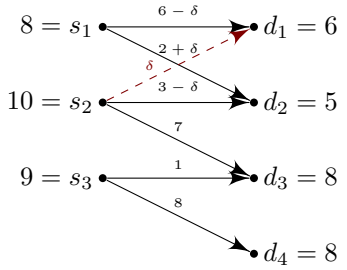
$$\lambda_i - \mu_i \leq c_{ij}$$

If it is satisfied everywhere, we have optimality. Otherwise, will have to do something.

What we do is we add an edge, say from the second supplier to the first consumer. Then we have created a cycle. We keep increasing the flow on the new edge. This causes the values on other edges to change by flow conservation. So we keep doing this until some other edge reaches zero.

If we increase flow by, say, δ, we have

$6-\delta$	5	$2+\delta$	3		4		6
δ	2	$3-\delta$	7	7	4		1
	5		6	1	2	8	4



The maximum value of δ we can take is 3. So we end up with

3	5	5	3		4		6
3	2		7	7	4		1
	5		6	1	2	8	4

We re-compute the Lagrange multipliers to obtain

	-5		-3		-7		-9
					7		9
0	3	5	5	3		4	6
			0				6
-3	3	2		7	7	4	1
	0		-2				
-5		5		6	1	2	8 4

We see a violation at the bottom right. So we do it again:

3	5	5	3		4		6
3	2		7	$7-\delta$	4	δ	1
	5		6	$1+\delta$	2	$8-\delta$	4

The maximum possible value of δ is 7. So we have

3	5	5	3		4		6
3	2		7		4	7	1
	5		6	8	2	1	4

Calculating the Lagrange multipliers gives

	-5		-3		-2		-4	
					2		4	
0	3	5	5	3		4		6
			0		-1			
-3	3	2		7		4	7	1
	5		3					
0		5		6	8	2	1	4

No more violations. Finally. So this is the optimal solution.

5.4 The maximum flow problem

Suppose we have a network (V, E) with a single source 1 and a single sink n. There is no costs in transportation, but each edge has a capacity. We want to transport as much stuff from 1 to n as possible.

We can turn this into a minimum-cost flow problem. We add an edge from n to 1 with -1 cost and infinite capacity. Then the minimal cost flow will maximize the flow from n to 1 as possible. So the same amount of stuff will have to flow from 1 to n through the network.

We can write this is problem as

$$\text{maximize } \delta \text{ subject to}$$

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j(j,i)\in E} x_{ji} = \begin{cases} \delta & i = 1 \\ -\delta & i = n \\ 0 & \text{otherwise} \end{cases} \qquad \text{for each } i$$

$$0 \le x_{ij} \le C_{ij} \quad \text{for each } (i,j) \in E.$$

Here δ is the total flow from 1 to n.

While we can apply our results from the minimum-cost flow problem, we don't have to do so. There are easier ways to solve the problem, using the *max-flow min-cut theorem*.

First we need to define a cut.

Definition (Cut)**.** Suppose $G = (V, E)$ with capacities C_{ij} for $(i, j) \in E$. A *cut* of G is a partition of V into two sets.

For $S \subseteq V$, the *capacity* of the cut $(S, V \setminus S)$ is

$$C(S) = \sum_{(i,j)\in(S\times(V\setminus S))\cap E} C_{ij},$$

All this clumsy notation says is that we add up the capacities of all edges from S to $V \setminus S$.

Assume x is a feasible flow vector that sends δ units from 1 to n. For $X, Y \subseteq V$, we define

$$f_x(X, Y) = \sum_{(i,j)\in(X\times Y)\cap E} x_{ij},$$

i.e. the overall amount of flow from X to Y.

For any solution x_{ij} and cut $S \subseteq V$ with $1 \in S, n \in V \setminus S$, the total flow from 1 to n can be written as

$$\delta = \sum_{i \in S} \left(\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} \right).$$

This is true since by flow conservation, for any $i \neq 1$, $\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = 0$, and for $i = 1$, it is δ. So the sum is δ. Hence

$$\begin{aligned}
\delta &= f_x(S, V) - f_x(V, S)\\
&= f_x(S, S) + f_x(S, V \setminus S) - f_x(V \setminus S, S) - f_x(S, S)\\
&= f_x(S, V \setminus S) - f_x(V \setminus S, S)\\
&\leq f_x(S, V \setminus S)\\
&\leq C(S)
\end{aligned}$$

This says that the flow through the cut is less than the capacity of the cut, which is obviously true. The less obvious result is that this bound is tight, i.e. there is always a cut S such that $\delta = C(S)$.

Theorem (Max-flow min-cut theorem)**.** Let δ be an optimal solution. Then

$$\delta = \min\{C(S) : S \subseteq V, 1 \in S, n \in V \setminus S\}$$

Proof. Consider any feasible flow vector x. Call a path $v_0, \cdots, v_k$ an *augmenting path* if the flow along the path can be increased. Formally, it is a path that satisfies

$$x_{v_{i-1}v_i} < C_{v_{i-1}v_i} \text{ or } x_{v_iv_{i-1}} > 0$$

for $i = 1, \cdots, k$. The first condition says that we have a forward edge where we have not hit the capacity, while the second condition says that we have a backwards edge with positive flow. If these conditions are satisfied, we can increase the flow of each edge (or decrease the backwards flow for backwards edge), and the total flow increases.

Now assume that x is optimal and let

$$S = \{1\} \cup \{i \in V : \text{ there exists an augmenting path from 1 to } i\}.$$

Since there is an augmenting path from 1 to S, we can increase flow from 1 to any vertex in S. So $n \notin S$ by optimality. So $n \in V \setminus S$.

We have previously shown that

$$\delta = f_x(S, V \setminus S) - f_x(V \setminus S, S).$$

We now claim that $f_x(V \setminus S, S) = 0$. If it is not 0, it means that there is a node $v \in V \setminus S$ such that there is flow from v to a vertex $u \in S$. Then we can add that edge to the augmenting path to u to obtain an augmenting path to v.

Also, we must have $f_x(S, V \setminus S) = C(S)$. Or else, we can still send more things to the other side so there is an augmenting path. So we have

$$\delta = C(S). \qquad \square$$

The max-flow min-cut theorem does not tell us *how* to find an optimal path. Instead, it provides a quick way to confirm that our path is optimal.

It turns out that it isn't difficult to find an optimal solution. We simply keep adding flow along augmenting paths until we cannot do so. This is known as the *Ford-Fulkerson algorithm*.

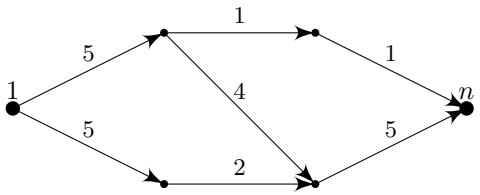
(i) Start from a feasible flow x, e.g. $x = \mathbf{0}$.

(ii) If there is no augmenting path for x from 1 to n, then x is optimal.

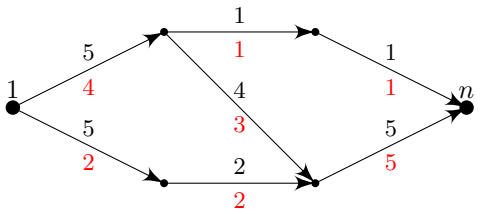
(iii) Find an augmenting path for x from 1 to n, and send a maximum amount of flow along it.

(iv) `GOTO` (ii).

Example. Consider the diagram



We can keep adding flow until we reach



(red is flow, black is capacity). We know this is an optimum, since our total flow is 6, and we can draw a cut with capacity 6: